Collective Memory Transfers for Multi-Core Chips

George Michelogiannakis, Alexander Williams,
Samuel Williams, John Shalf

Computer Architecture Laboratory
Lawrence Berkeley National Laboratory

International Conference on Supercomputing (ICS) 2014

In a Nutshell

- Future technologies will allow more parallelism on chip
- Computational throughput expected to increase faster than memory bandwidth
  - Pin and power limitations for memory
- Many applications are limited by memory bandwidth
- We propose a mechanism to coordinate memory accesses between numerous processors such that the memory is presented with in-order requests
  - Increases DRAM performance and power efficiency

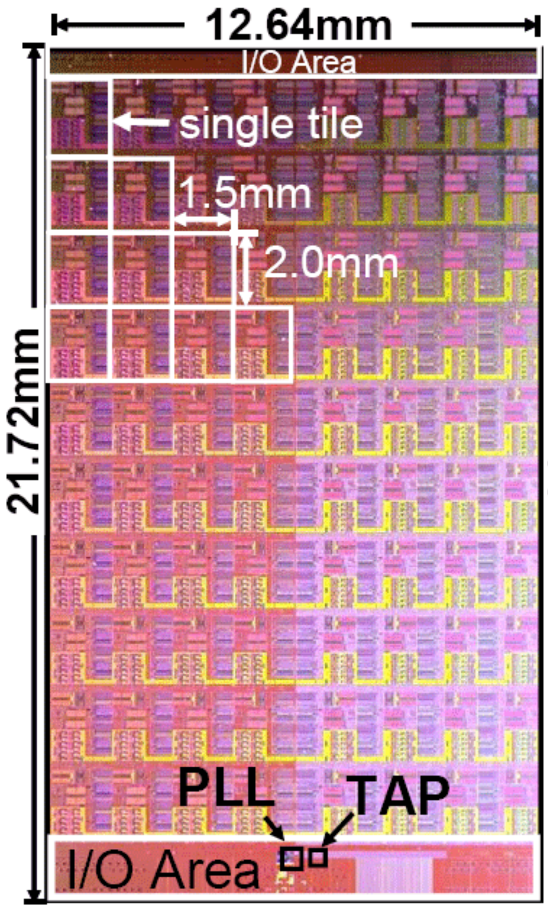
Today's Menu

- Today's and future challenges
- The problem
- Collective memory transfers
- Evaluation
- Related work, future directions and conclusion

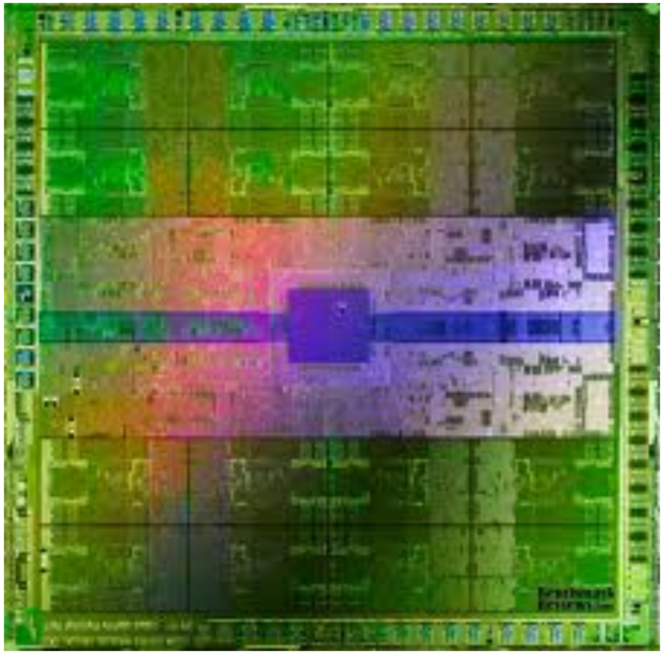
Chip Multiprocessor Scaling

By 2020 we may witness 2048-core chip multiprocessors

Intel 80-core



NVIDIA Fermi:
512 cores

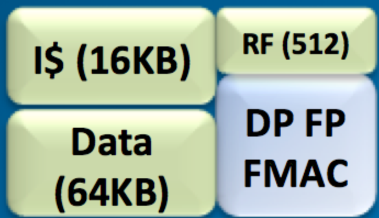


AMD Fusion:
four full CPUs
and 408 graphics
cores

Straw-man Exascale Processor

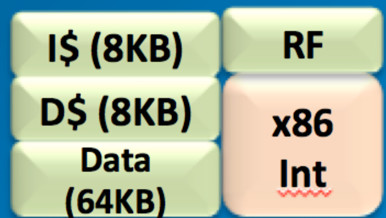
Shekhar Borkar, 2014

Execution (Exe) Function



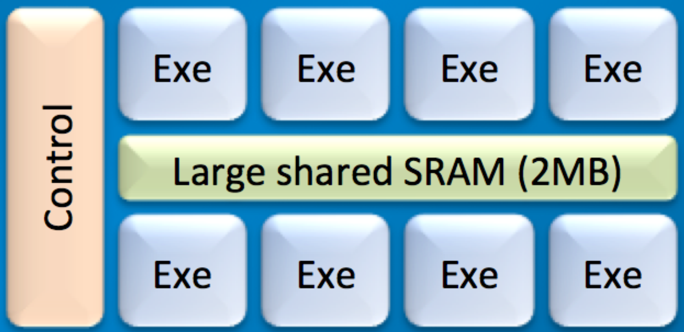
Application specific

Control Function

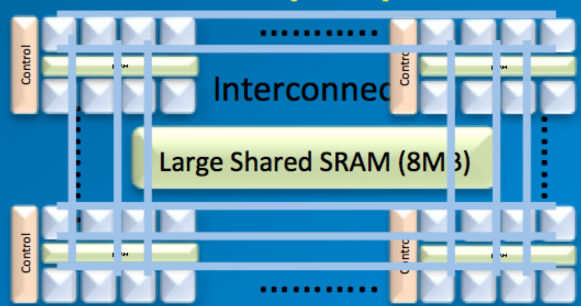


System SW

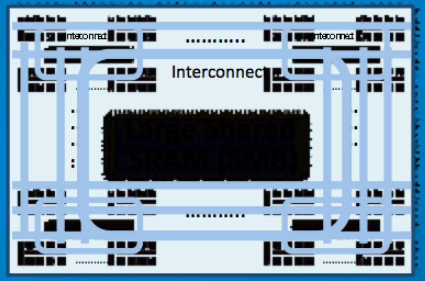
8 Exe + Control



Cluster (16 x)

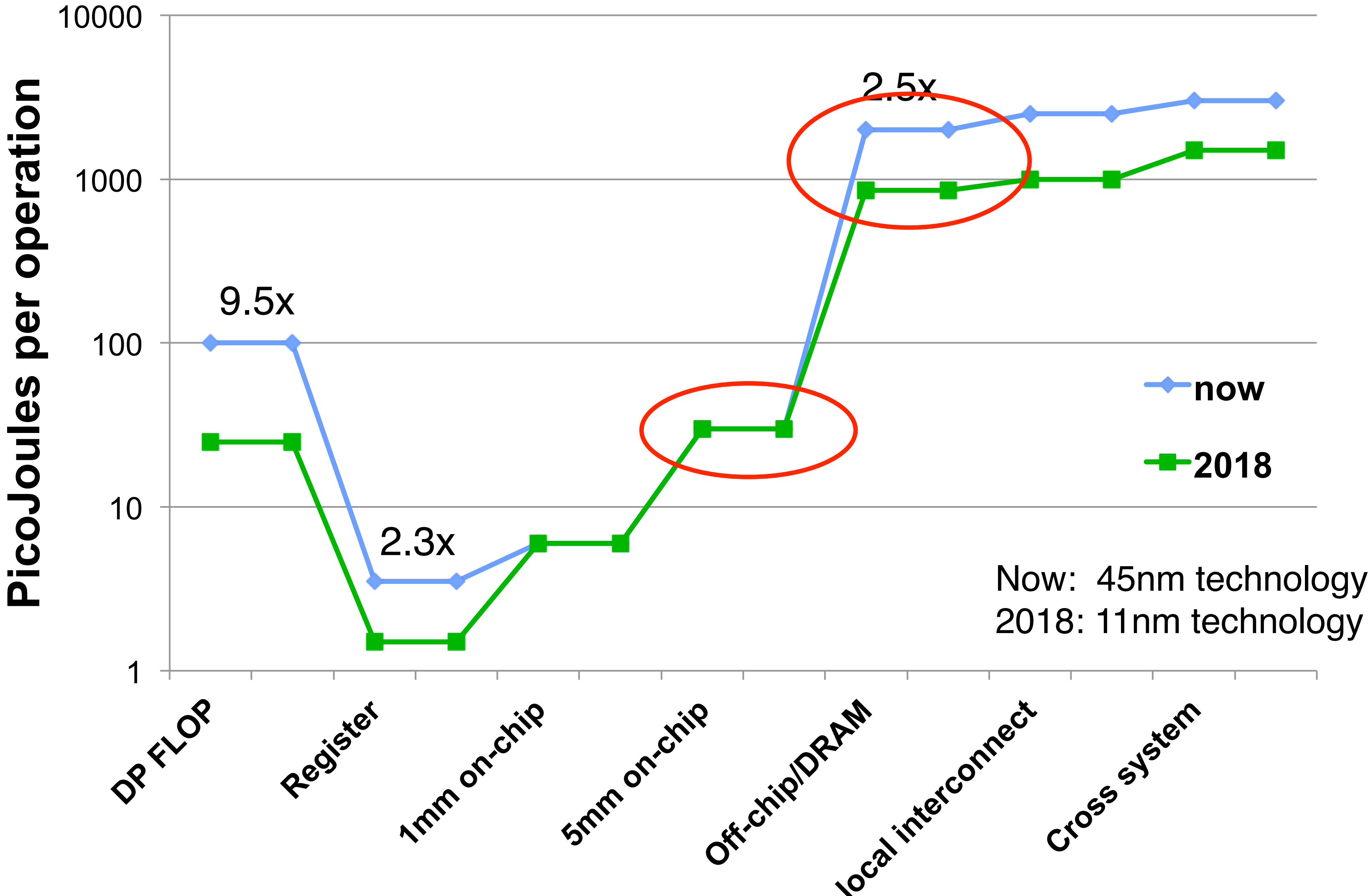


Processor Chip (~16 Clusters)



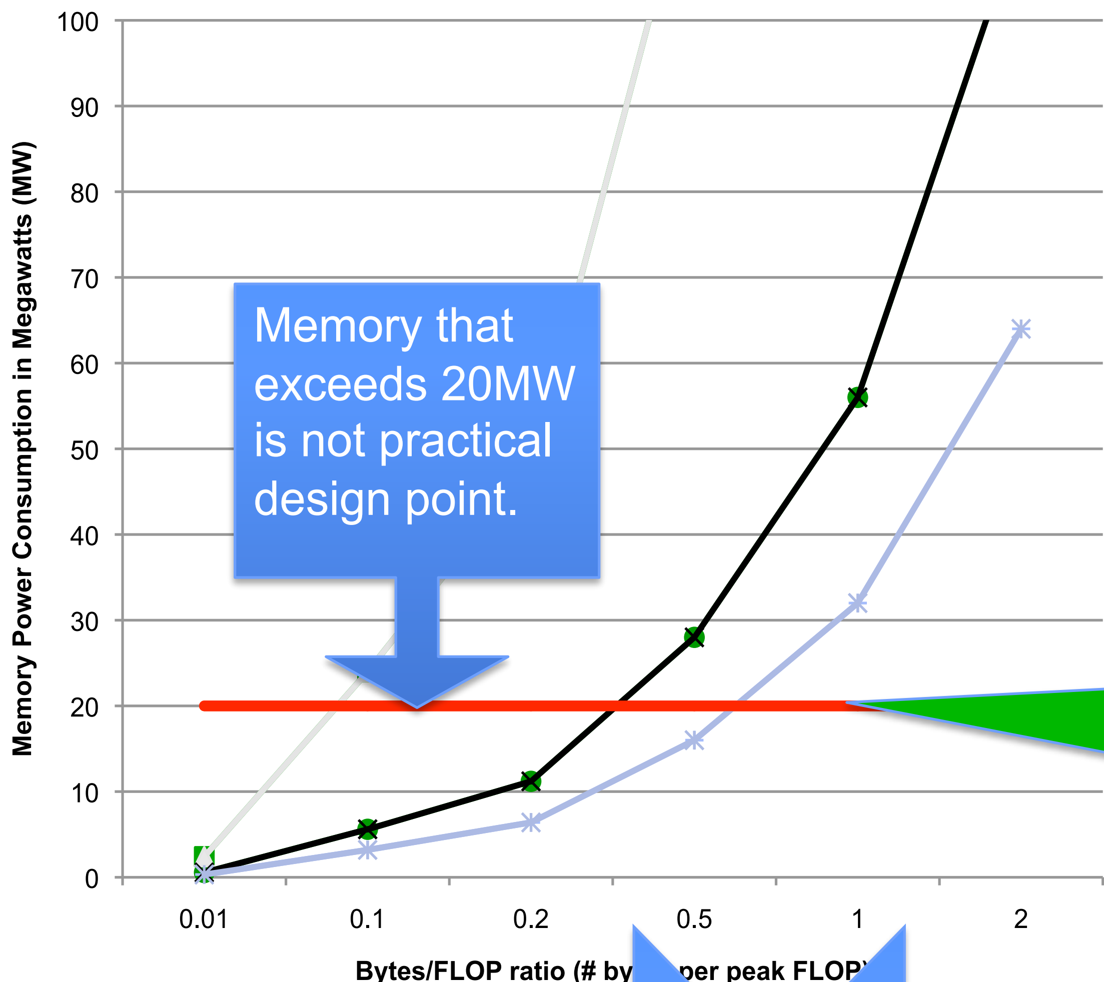
Technology	4nm, 2020
Die area	16x16 mm2
Cores/die	2000
Frequency	1.1 GHz@Vdd
TFLOPs	4 TF Peak@Vdd
Power	15 W@Vdd
E Efficiency	4 pJ/F@Vdd, much better at NTV

Data Movement and Memory Dominate



Exascale computing technology challenges. VECPAR 2010

Memory Bandwidth a Constraint



Wide variety of applications are memory bandwidth bound

Memory that exceeds 20MW is not practical design point.

Memory Technology Investment enables improvement in bandwidth (and hence improves application breadth)

Application performance and breadth pushes us to higher BW

Power pushes us to lower bandwidth

Exascale computing technology challenges. VECPAR 2010

Therefore…

- Parallelism will increase
- Compute capacity increases faster than memory bandwidth
  - 10% memory bandwidth increase per year [1]
  - Compute capacity increase driven by Moore’s law
- Data movement and memory access power already a limiting factor
  - Projected to worsen with future technologies
- Numerous applications are memory bandwidth bound
  - Will become worse in the future

[1] Scaling the bandwidth wall: challenges in and avenues for CMP scaling. ISCA 2009

Today’s Menu

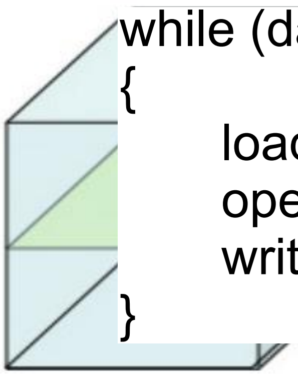
- Today’s and future challenges
- The problem
- Collective memory transfers
- Evaluation
- Related work, future directions and conclusion

Computation on Large Data

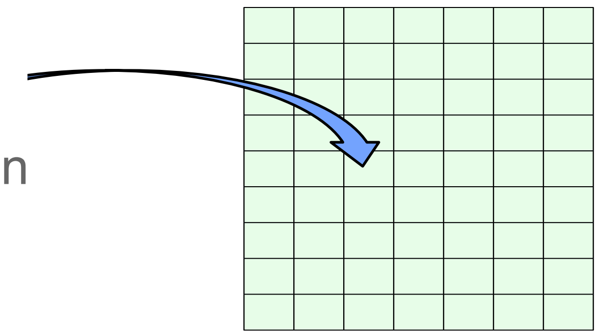
3D space
Slice into 2D planes

2D plane for center of stencil
still too large for single
processor

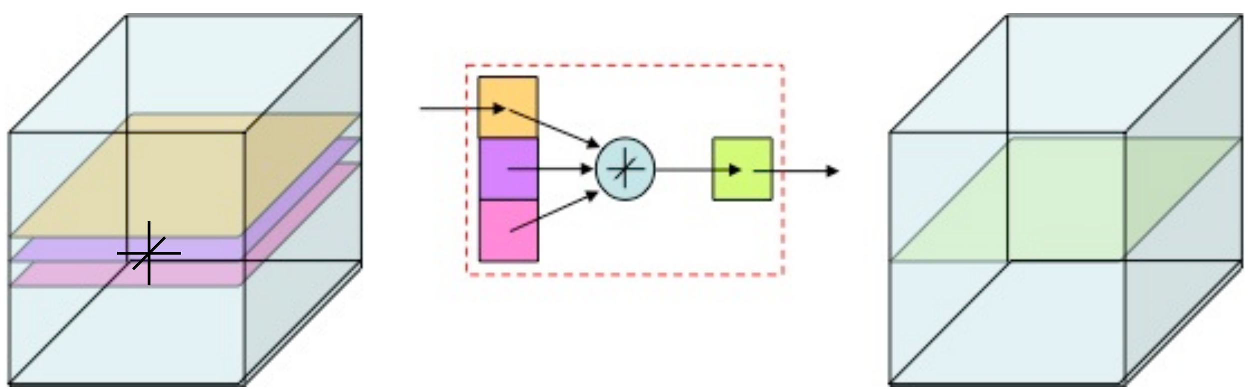
Divide array into tiles
One tile per processor
Sized for L1 cache



```
while (data_remaining)
{
        load_next_tile(); // DMA load
        operate_on_tile(); // Local computation
        write_resulting_tile(); // DMA write
}
```

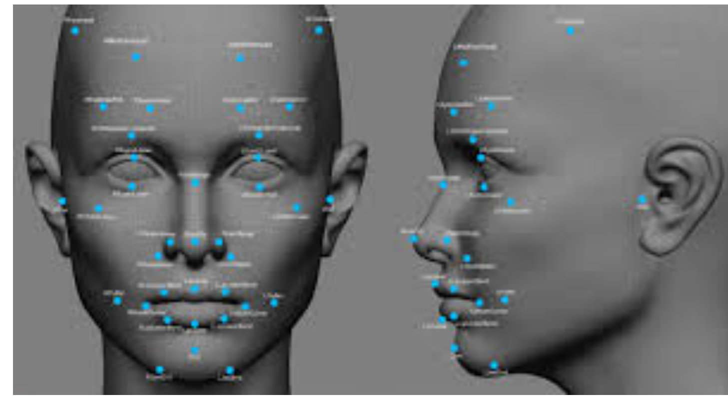
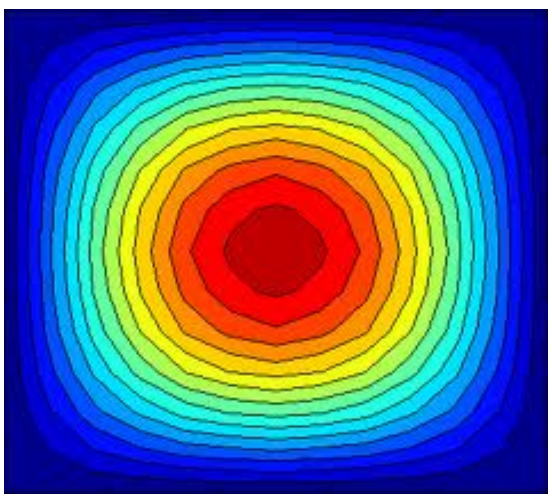
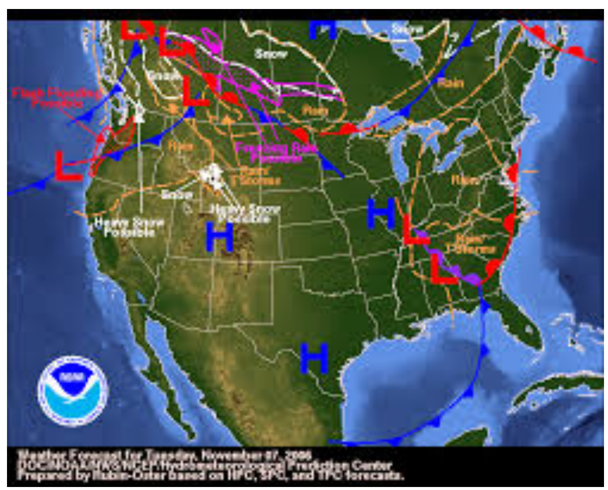
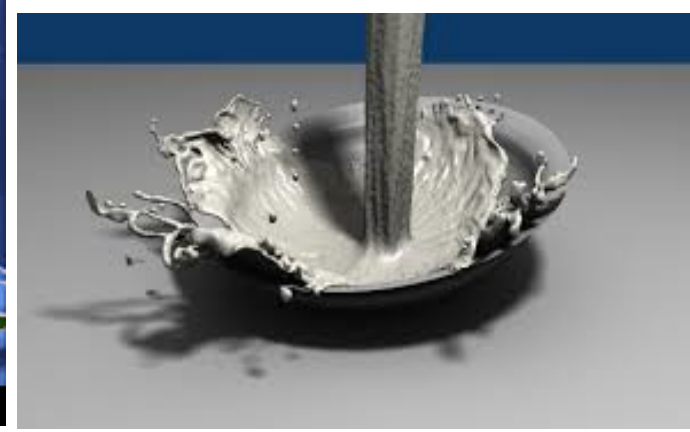


Full 3D Generalization



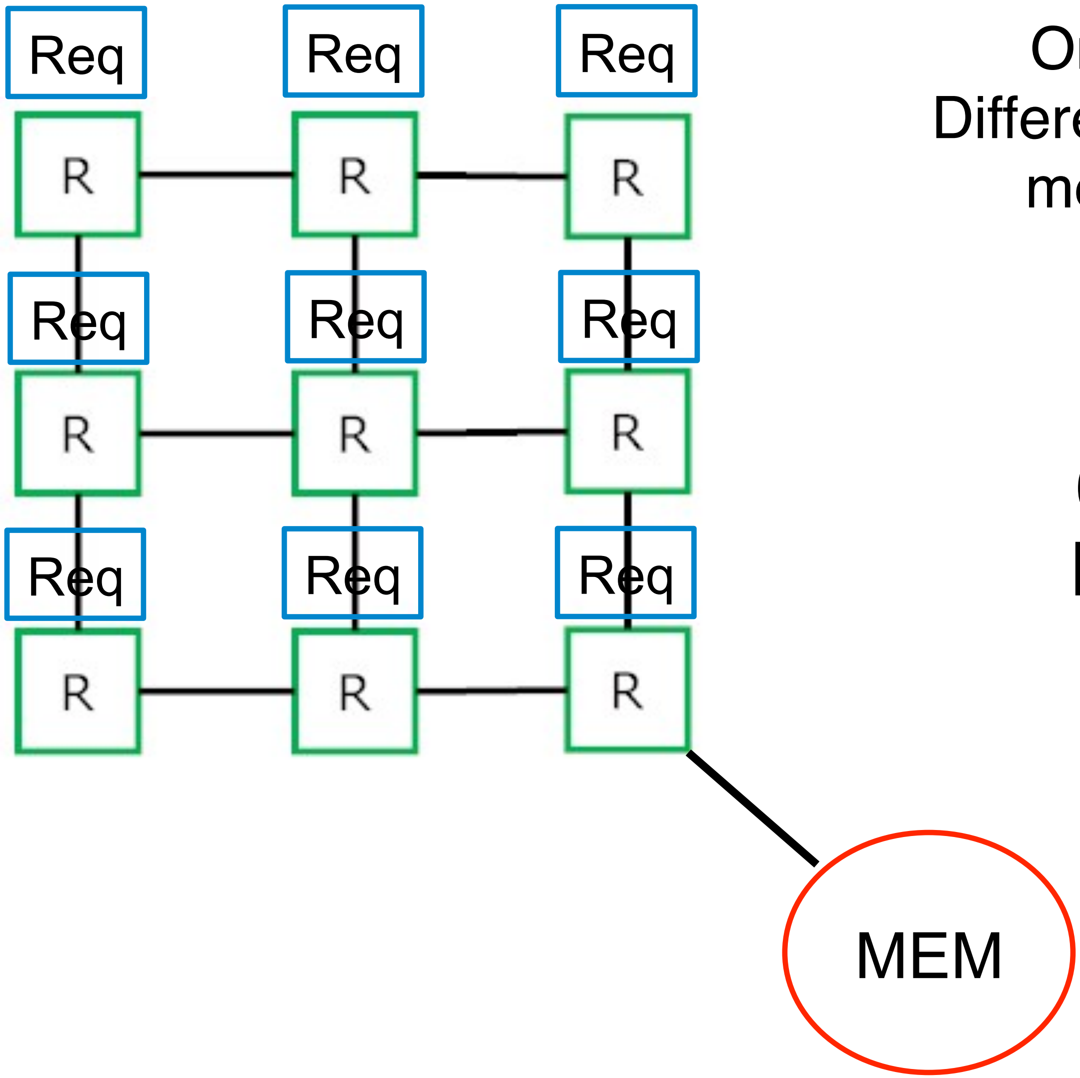
Data-Parallelism Covers a Broad Range of Applications

- From HPC to embedded computing
- Data-parallel applications a major driver for multi-cores

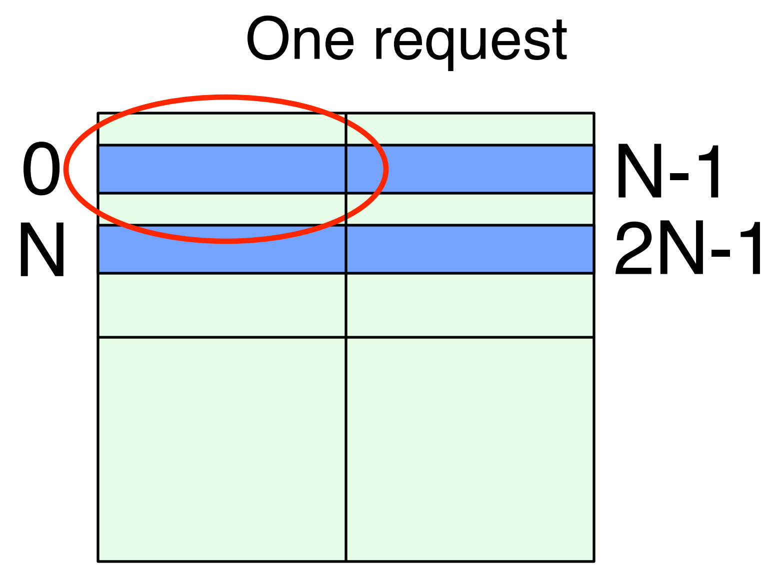






Convergence of recognition, mining, and synthesis workloads and its implications. Proc. IEEE 2008

The Problem: Unpredictable and Random Order Memory Access Pattern

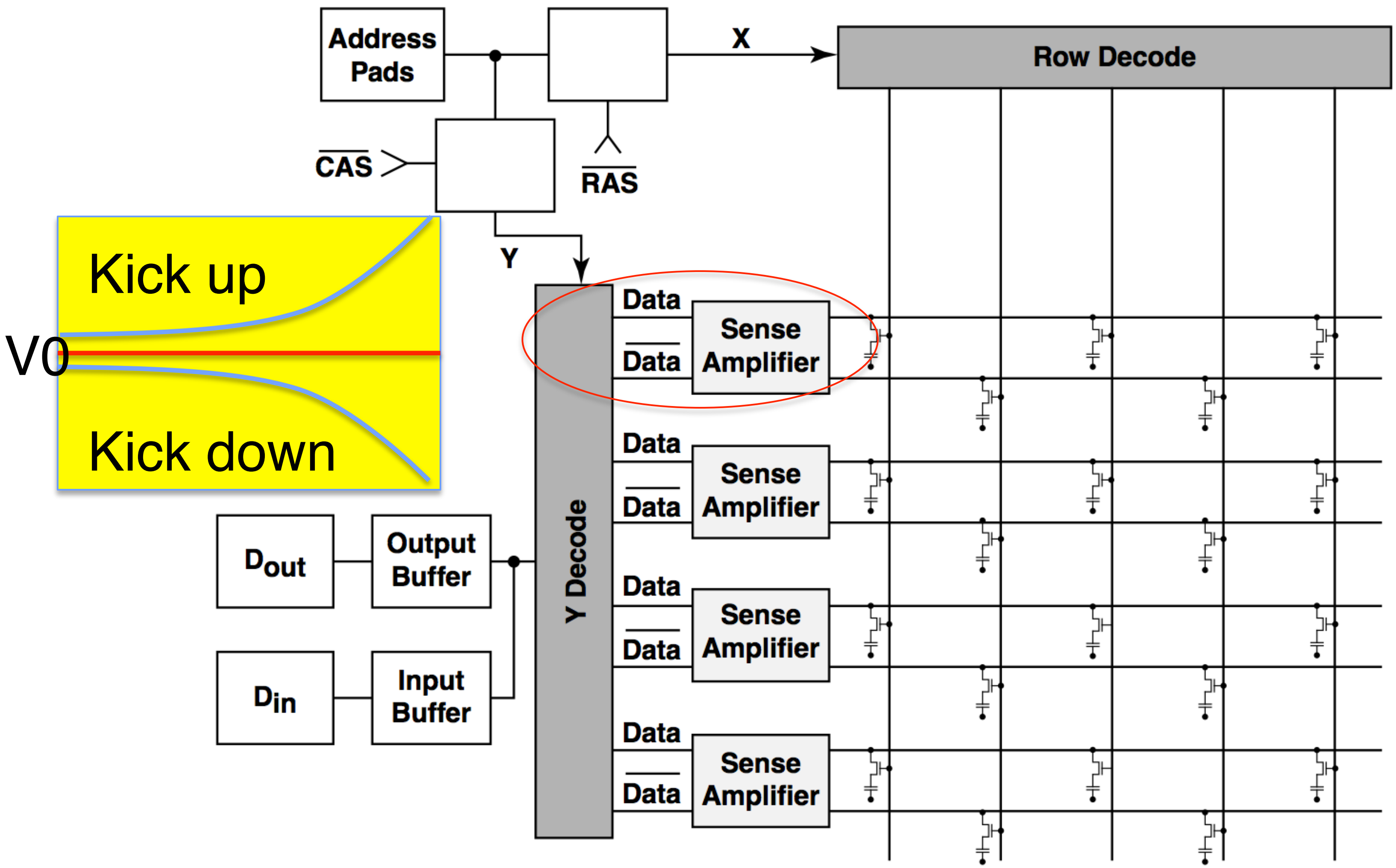


One request per *tile line*
Different tile lines have different memory address ranges



Row-major mapping

This is a DRAM Array

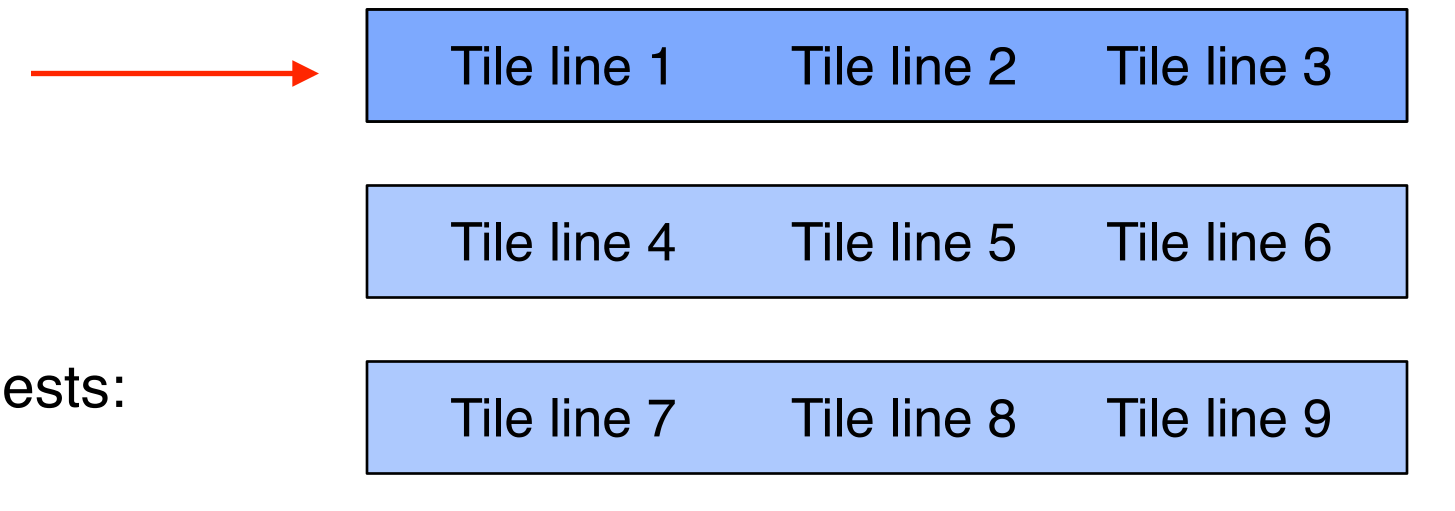


Source: ICE, "Memory 1997"

22430

Random Order Access Patterns Hurt DRAM Performance and Power

Reading tile 1 requires row activation and copying



In order requests:
3 activations

Worst case:
9 activations

Impact

- DRAMSim2 [2] with simple in-order and out-of-order traces
  - A single request accesses one 64-Byte word
  - FRFCFS memory scheduler
  - 16MB DDR3 Micron memory module
- DRAM throughput drops 25% for loads and 41% for stores
- Median latency increases 23% for loads and 64% for stores
- Power increases by 2.2x for loads and 50% for stores

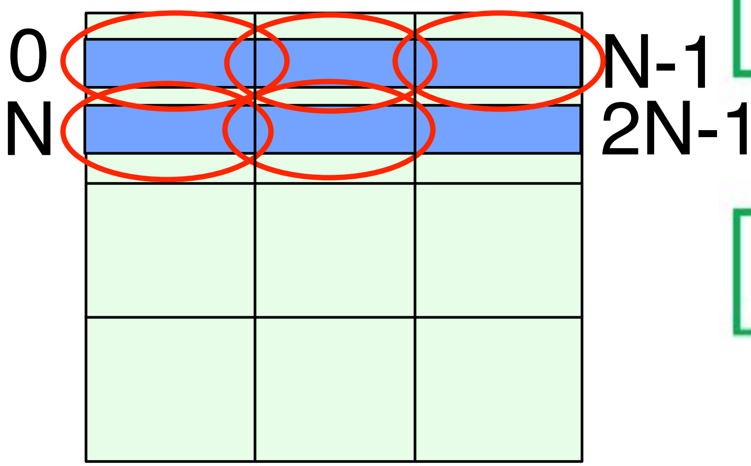
[2] DRAMSim2: A cycle accurate memory system simulator. IEEE CAL 2011

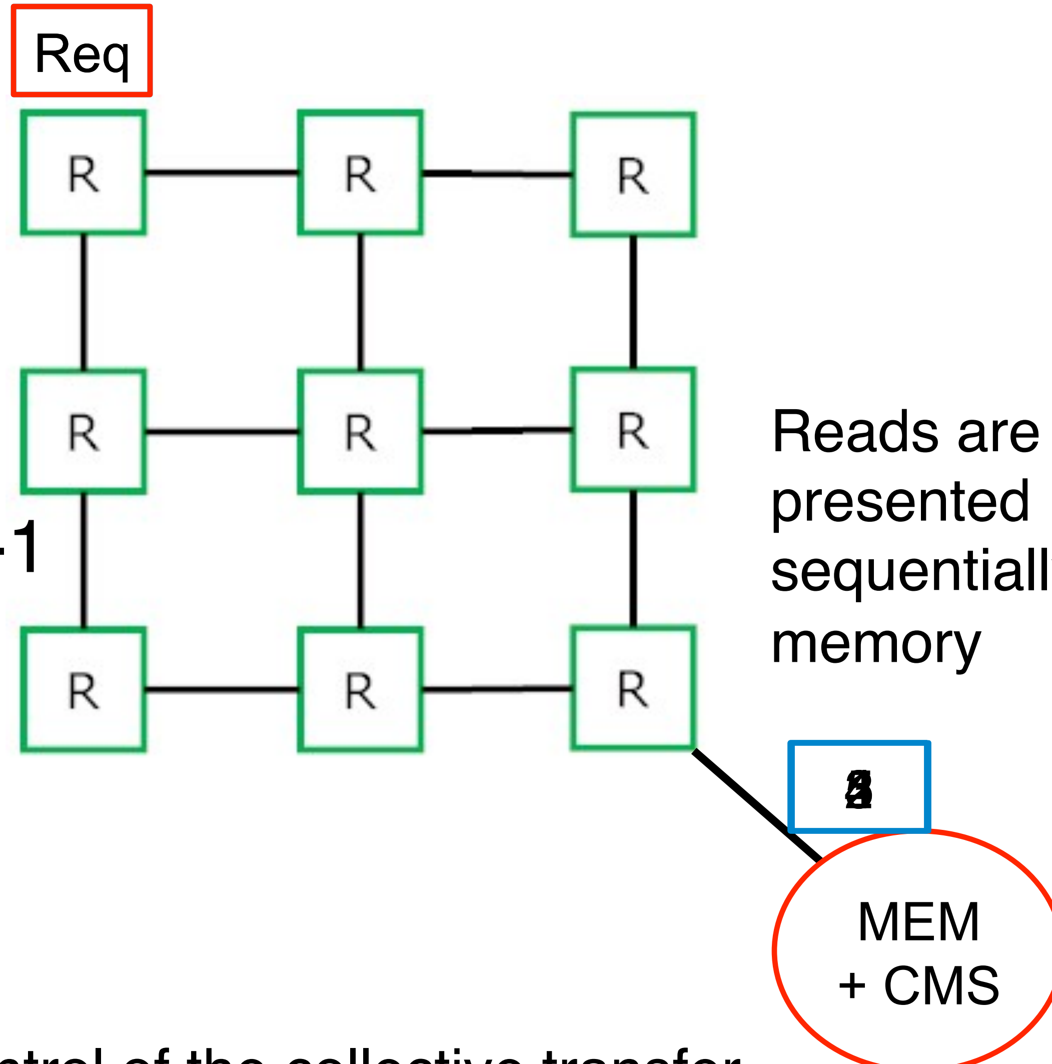
Today’s Menu

- Today’s and future challenges
- The problem
- Collective memory transfers
- Evaluation
- Related work, future directions and conclusion

Collective Memory Transfers

Requests replaced with one *collective request* on behalf of all processors



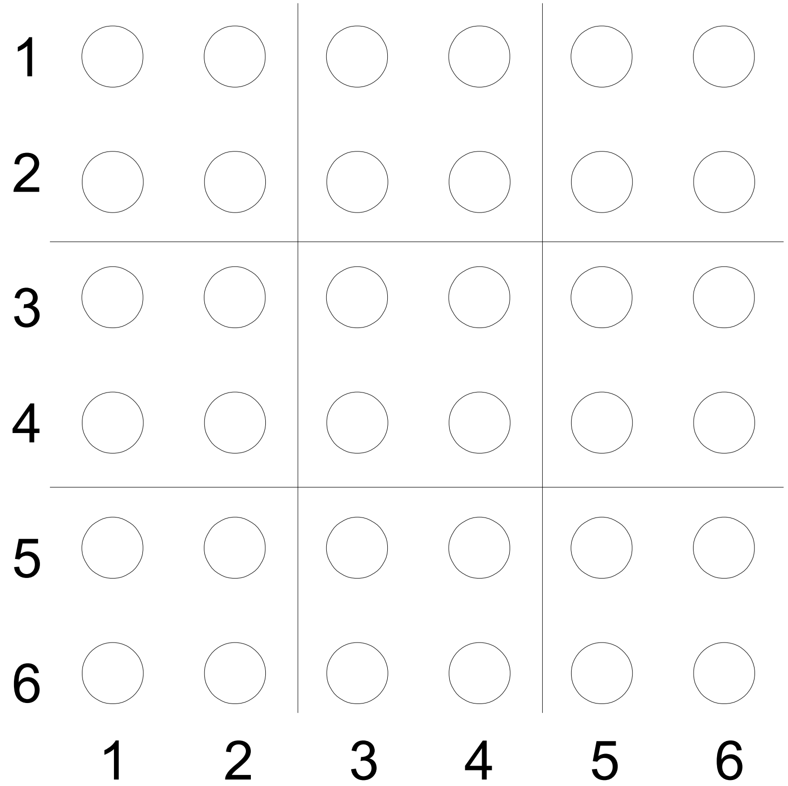


Reads are presented sequentially to memory

The CMS engine takes control of the collective transfer

Hierarchical Tiled Arrays to Transfer Data Layout Information

```
Array = hta(name,
        {[1,3,5], // Tile boundaries before
                  //  rows 1 (start),3 and 5
         [1,3,5]},// Likewise for columns
         [3,3]);  // Map to a 3x3 processor array
```



“The hierarchically tiled arrays programming approach”. LCR 2004

Hierarchical Tiled Arrays to Transfer Data Layout Information

Array = hta(name, {[1,3,5],[1,3,5]}, [3,3],
F(x) = x); // Mapping function or matrix

Loading a HTA with a CMS read

HTA_instance = CMS_read (HTA_instance);

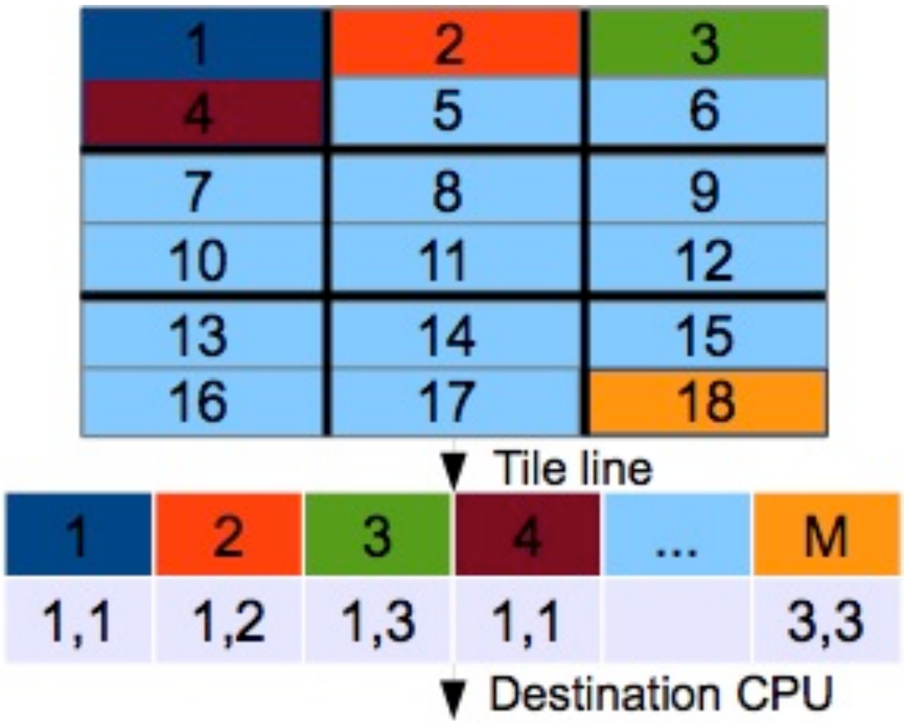
Loading the same HTA with DMA operations for each line of data

Array[row1] = DMA (Starting_address_row1,
Ending_address_row1);
.
.
Array[rowN] = DMA (Starting_address_rowN,
Ending_address_rowN);

“The hierarchically tiled arrays programming approach”. LCR 2004

Irregular Data Array Mappings

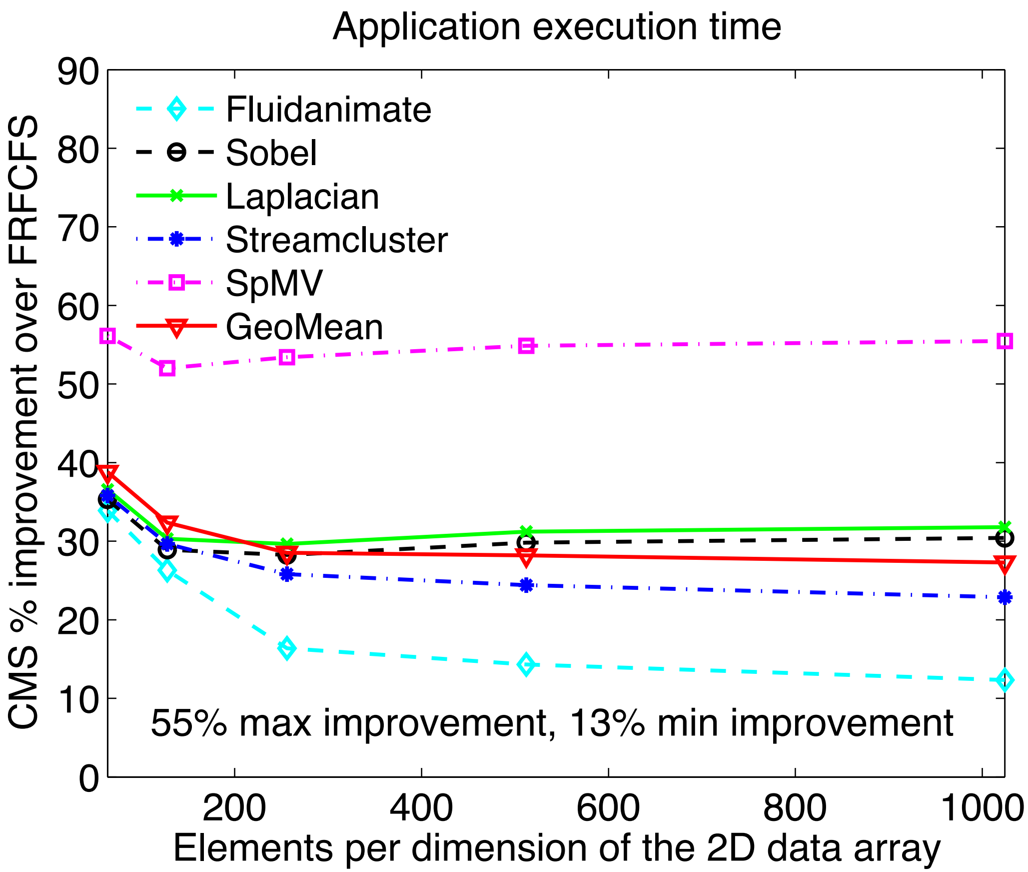
- If data array is not tiled, transferring the layout information over the on-chip network is too expensive
- Instead, the CMS engine learns the mapping by observing each processor’s requests in the first iteration of the application’s loop



Today's Menu

- Today's and future challenges
- The problem
- Collective memory transfers
- Evaluation
- Related work, future directions and conclusion

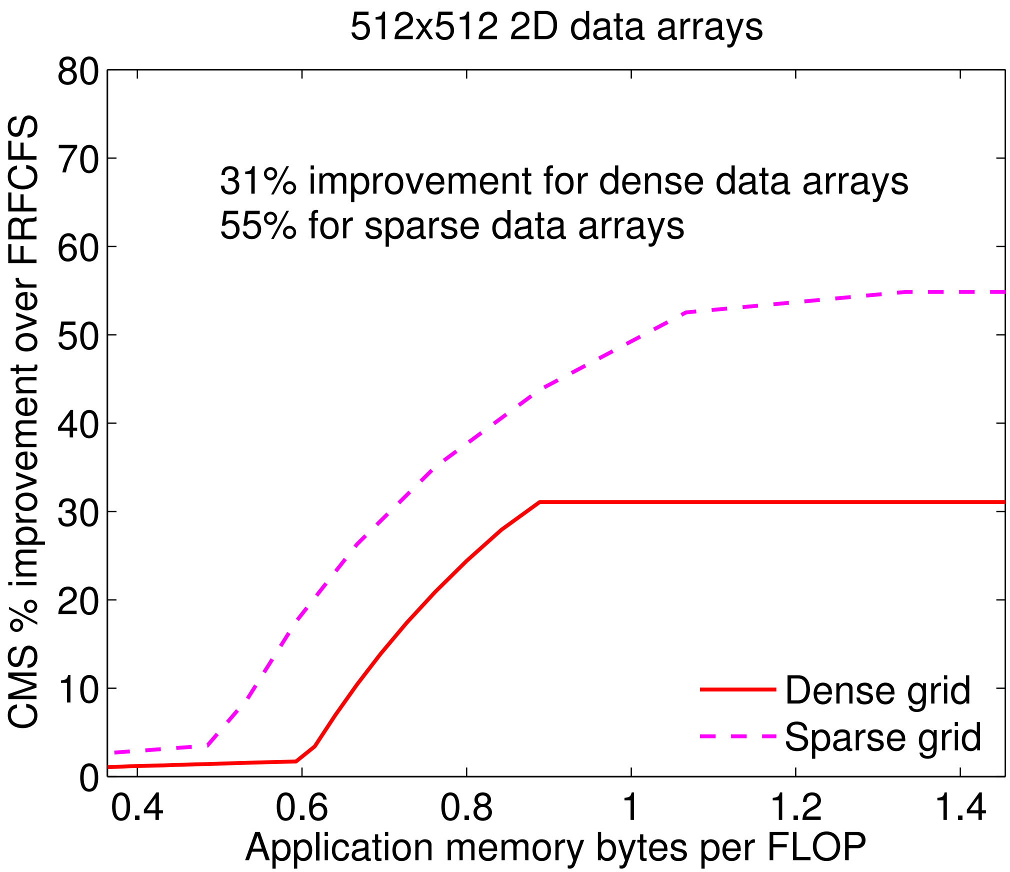
Execution Time Impact



8x8 mesh (64 CPUs)
Four memory controllers
Micron 16MB 1600MHz modules with a 64-bit data path
Xeon Phi processors

- Up to 55% application execution time reduction due to memory b/w
  - 27% geometric mean

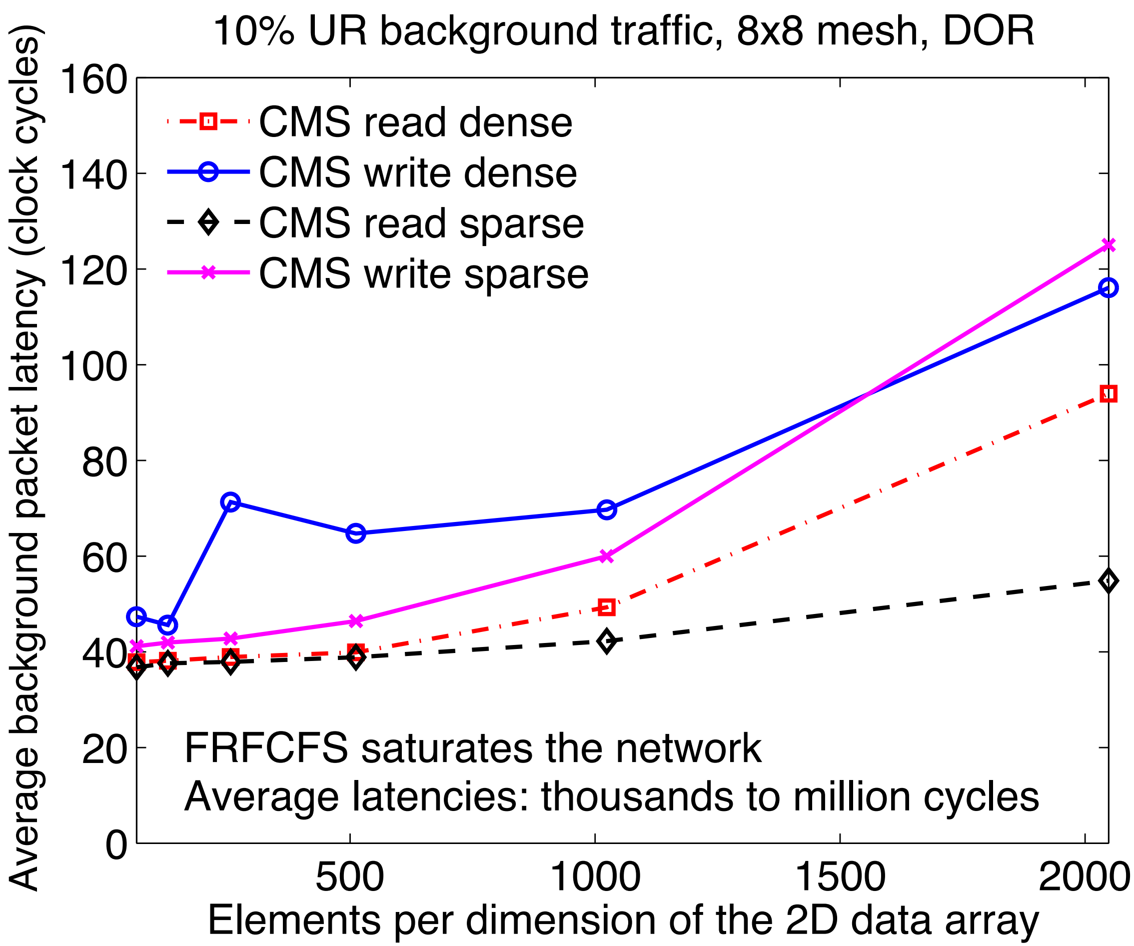
Execution Time Impact



8x8 mesh (64 CPUs)
Four memory controllers
Micron 16MB 1600MHz modules with a 64-bit data path
Xeon Phi processors

- 31% improvement for dense grid applications. 55% for sparse
- Sparse grid applications have lower computation times therefore they exert more pressure to the memory

Relieving Network Congestion



CMS Engine Implementation

CMS significantly simplifies the memory controller because shorter FIFO-only transaction queues are adequate

ASIC Synthesis	DMA	CMS
Combinational area (μm^2)	743	16231
Non-combinational area (μm^2)	419	61313
Minimum cycle time (ns)	0.6	0.75

To offset the cycle time increase, we can add a pipeline stage (insignificant effect compared to the duration of a transaction)

Today's Menu

- Today's and future challenges
- The problem
- Collective memory transfers
- Evaluation
- Related work, future directions and conclusion

Related Work

- A plethora of memory controller schedulers
  - However, the majority are passive policies that do not control the order requests arrive to the memory controller
  - Can only choose from within the transaction queue
- LLCs can partially re-order writes to memory
  - Write-through caches preferable in data-parallel computations [3]
  - CMS focuses on fetching new data and writing old data
- Prefetching focuses on latency, not bandwidth
  - Mispredictions are possible
  - Lacks application knowledge
- Past work uses injection control [4] or routers to partially re-order requests [5]

[3] Stencil computation optimization and auto-tuning on state-of-the-art multicore architectures. SC 2008
[4] Entry control in network-on-chip for memory power reduction. ISLPED 2008
[5] Complexity effective memory access scheduling for many-core accelerator architectures. MICRO 2009

Ongoing and Future Work

- What is the best interface to CMS from the software?
  - A library with an API similar to DMA function calls (the one shown)?
  - Left to the compiler to recognize collective transfers?

- How would this work with hardware-managed cache coherency?
  - Prefetchers may need to recognize and initiate collective transfers
  - Collective prefetching?
  - How to modify MESI to support force-feeding data to L1s

Conclusions

- Memory bandwidth will be an increasing limiting factor in application performance
- We propose a software-hardware collective memory transfer mechanism to present the DRAM with in-order accesses
  - Cores access the DRAM as a group instead of individually
- Up to 55% application execution time increase
  - 27% geometric mean

Questions?